設計段階での計算機性能予測システムの構想

西 正明* ・ 林川 基治**

The Plan for the Computer Performance Prediction System in a Design Stage

Masaaki NISHI* , Motoharu HAYASHIKAWA**

(Received September 28, 2001)

1. まえがき

現在さまざまな分野で計算機が利用されており，特に設計の分野では計算機の支援なしでは成り立たない。計算機の設計においても論理の自動生成[1]など，計算機自体の設計にも計算機を用いた細かい設計手順が施されるようになっている。計算機の設計は，一般に目標性能を予め定めて着手する。その際に必要になるのが，設計段階でのできるだけ正確な性能予測の手立てである。

計算機の性能はハードウェアの面からは，マシンサイクルタイムがその指標となる。計算機ハードウェア技術の開発においては，どのような計算機ハードウェアを開発すればどの程度のマシンサイクルタイムを達成できるのかを，ハードウェアの方式検討段階で，予め予測評価できることが必要である。計算機の性能を正確に予測するためには，計算機の論理設計と整合をとったハードウェアの開発ができるように，計算機論理と一体化したマシンサイクルタイム予測が必要である。

本論文では，計算機の設計段階で，さまざまな計算機実装方式と論理方式での性能を予測するシステムの構想を試みる。

2. 性能予測システムの構想

2.1 全体構成

方式検討段階でのマシンサイクルタイムの予測評価を実現するために，本性能予測システムを図1に示すように構成することにした。各段階での処理結果がディレー要素ファイルに保存されるようにする。各処理の内容を以下に述べる。

①ディレー要素生成

計算機の論理動作記述から，ハードウェア部品とその接続関係を示す論理構造の記述を生成する。これは最終的にディレー値を求める際の基になるものである。

* 信州大学教育学部 ** 山口大学教育学部

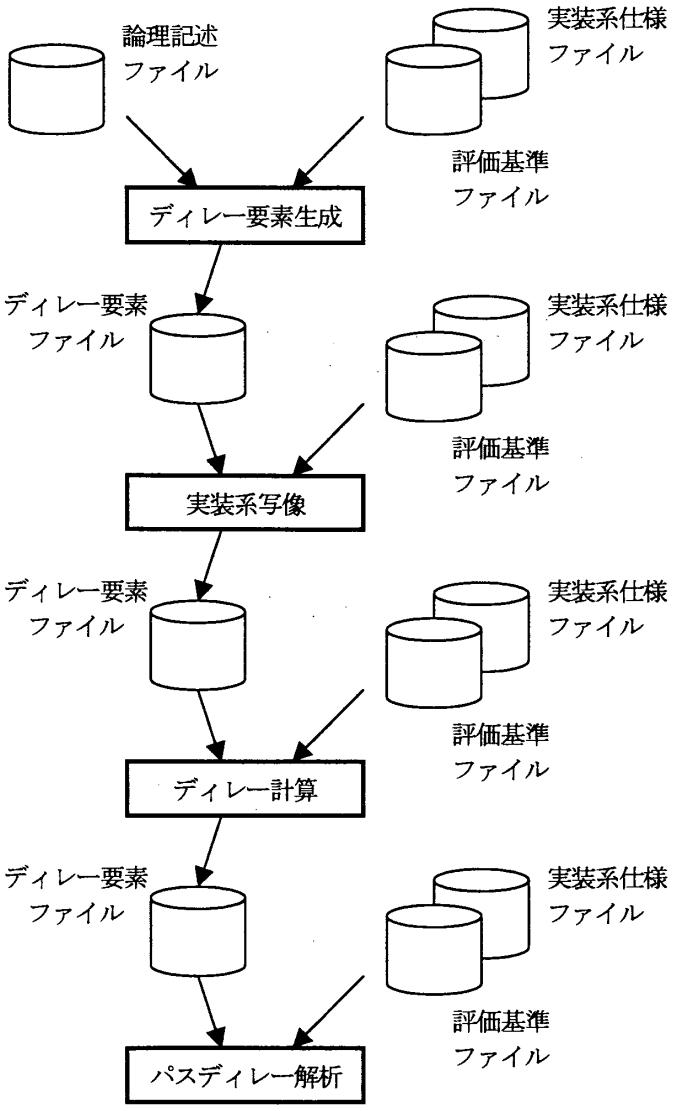

図 1　性能予測システムの構成

②実装系写像

生成した論理構造を，モジュールや LSI などに分割し，配置を決定する。

③ディレー計算

分割配置後の論理構造に対してクリティカルパスを探索し，そのパスディレーを計算する。

④パスディレー解析

パスディレーの統計解析を行い，ディレーが最大のパスのディレー値を求め，マシンサイクルタイムの予測値とする。

次に，図 1 におけるシステム構成の入出力ファイルについて述べる。

①論理記述ファイル

計算機の論理を記述したファイルである。論理動作をレジスタトランスファレベルで記述する。計算機の論理を，まず全体を 1 つの部品として定義する。この部品は，入出力信号と，

幾つかの下位階層の部品と論理動作記述を含む。下位階層の部品はさらに下位の階層の部品を含むというように階層的に記述する。このファイルによって，論理方式を記述して評価する。

②実装系仕様ファイルと評価基準ファイル

実装系仕様ファイルはLSI，モジュール，ボードなどのハードウェア部品のサイズ，配線層数，部品搭載位置などの分割配置に必要な機械的仕様と特性インピーダンス，単位長あたりの配線ディレー時間，配線容量などのディレー計算に必要な電気的仕様を記述する。

評価基準ファイルは配線長を推定するためのパラメータやディレーばらつきの計算に必要なパラメータ，クロックの周期・相数などを記述する。これらのファイルによって，実装方式を記述して評価する。

③ディレー要素ファイル

ディレー要素ファイルは論理記述ファイルからディレー要素生成の処理時に生成される。実装系写像での分割配置，ディレー計算の処理時には，ディレー要素ファイルを入力データとして使用し，同時にそのときに新たに得られる実装情報の出力先としても使用する。このようにして，処理の進行とともにディレー要素ファイルに処理結果を追加していく。

2.2 課題と方策

性能予測システムの課題とその対処法について述べる。本システムはマシンサイクルタイムの予測を目的としているため，膨大な数の論理回路で成るインストラクションプロセッサ全体を対象として，前節で述べた各処理を行なわなければならない。実装系写像，ディレー計算，クリティカルパス探索などの処理は，対象とするゲートの組合せ数に応じて，処理時間が増大する。本論文では，この処理時間の問題を以下の2つの方法で対処するようにした。

(1) 論理分割

全体を扱うのでなく，独立に扱えるような幾つかに分割する。どう分割するかが問題になるが，ここでは論理機能のまとまりに対応して分割することにする。ディレー要素ファイルで論理構造を階層的に記述することで，ボード，モジュール，LSIなどの現実の実装階層に限らず，仮想的にも自由に分割することができる。全体を対象にするのではなく，常に下位階層の部品を対象として論理分割すればよく，処理時間を短縮できる。

(2) 精粗混在記述

論理設計では全てを従来とは全く別物にすることは少なく，既存の論理をそのまま利用することが多い。したがって，既存と同じ部分については既にゲート数やパスディレー値がわかっている。これらの情報はそのまま利用できる。そこで，既知の論理構造についてはゲート数やパスディレー値だけを記述する粗記述，新しい論理記述部分については論理構造そのものの精密な記述として，両者の記述を混在させることにした。

図2にファイルの階層構成を示す．計算機はまず全体を1つの部品として記述する。記述項目は，入出力信号と幾つかの下位階層の部品，それらの接続関係および実装情報である。図2では部品CPU1があり，その下位部品CPU1.Aが同様の記述で展開されることを示している。論理記述ファイルでも，論理記述を論理機能のまとまりごとに部品として記述する。既知の論理構造については，入出力信号とゲート数やパスディレー値などの実装情報だけを

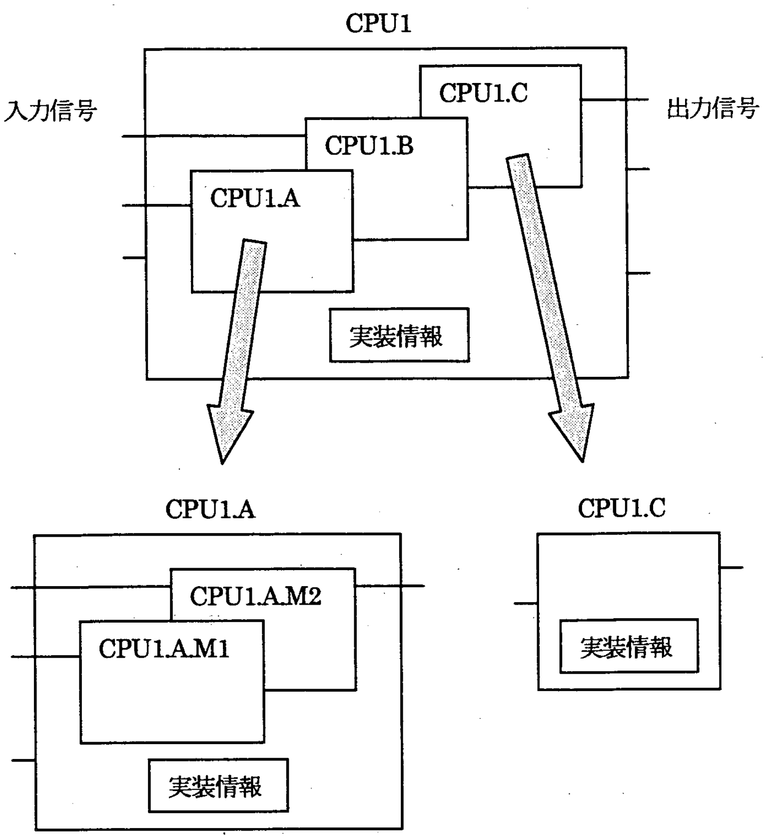

図 2　ファイルの階層構成

```
(ID (Delaynet クラス名)
    (Input   信号名 信号名 ・・・)
    (Output 信号名 信号名 ・・・)
    (Parts （インスタンス名
              (Class クラス名)
              (Setinput （下位部品信号名 上位部品信号名)
                  ・・・・・・・・・・・・・・・            )
              (Setoutput (下位部品信号名 上位部品信号名)))
           (インスタンス名
                  ・・・・・・・・・・・・・・・・・・・・・・))
    (Volume (使用ゲート数)
            (使用ピン数))
    (Delay （パス名 パスディレー値)
                  ・・・・・・・・・・・・・・・・・・・・・・))
```

図 3　ファイルのフォーマット

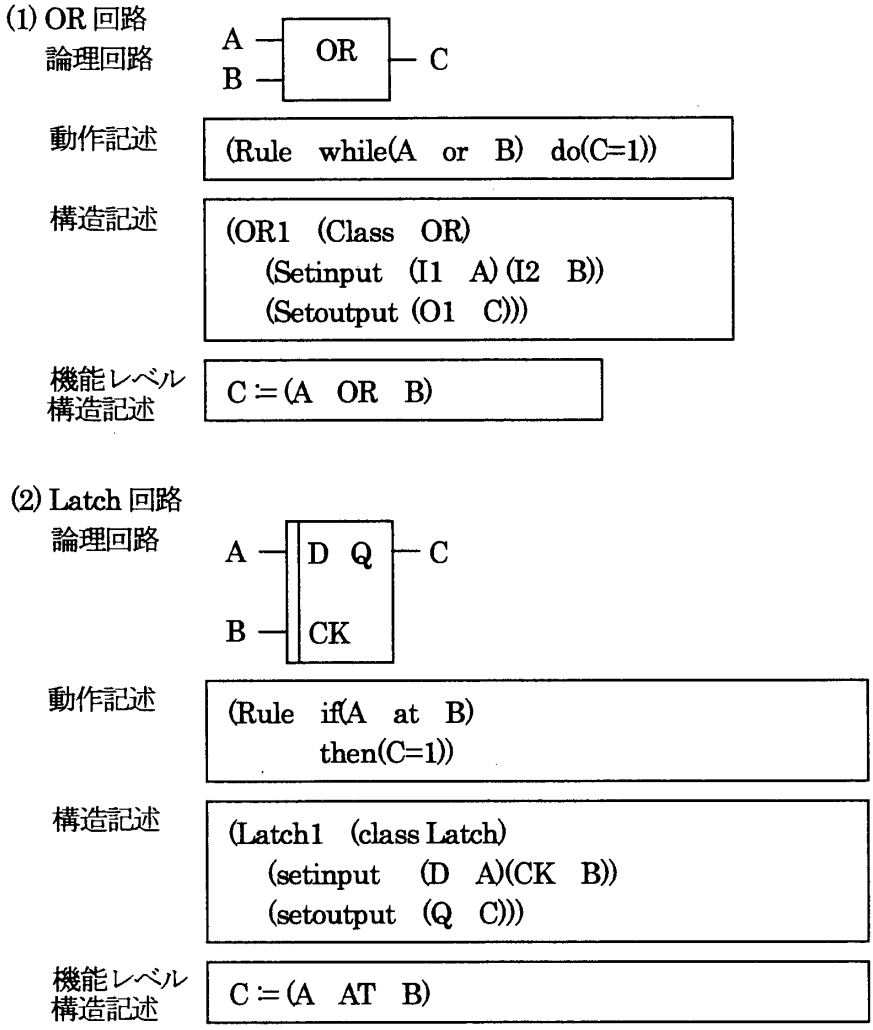

図 4 論理記述方法の一例

記述する粗記述でよく，図 2 の部品 CPU1.C がその例である。

図 3 にファイルのフォーマットを示す。入出力信号を Input，Output で示し，接続関係を Parts で示す。Volume と Delay で実装情報を記述する。実装情報は実装系写像とディレー計算の処理で書き加えられる。構成内容ごとにクラス名を対応させ，使用箇所ごとにインスタンス名を対応させることで，同じ構成内容を重複して記述するのを省いている。

3. ディレー要素生成

論理記述ファイルからディレー要素ファイルを生成するには，基本的には論理動作記述からそのまま構造記述に変換していけばよい。しかし，後述するように，動作記述に重複があると，構造記述が別個に生成されて無駄が生じる。それで本論文では，まず論理記述ファイルの動作記述から機能レベル構造記述を生成し，次に実装レベルの構造記述に変換することにする。こうすることで，構造記述の重複回避と論理の簡単化が同時に行われる。また，使用する実装系を変更する場合に，機能レベル構造記述までの処理をやり直さなくて済むよう

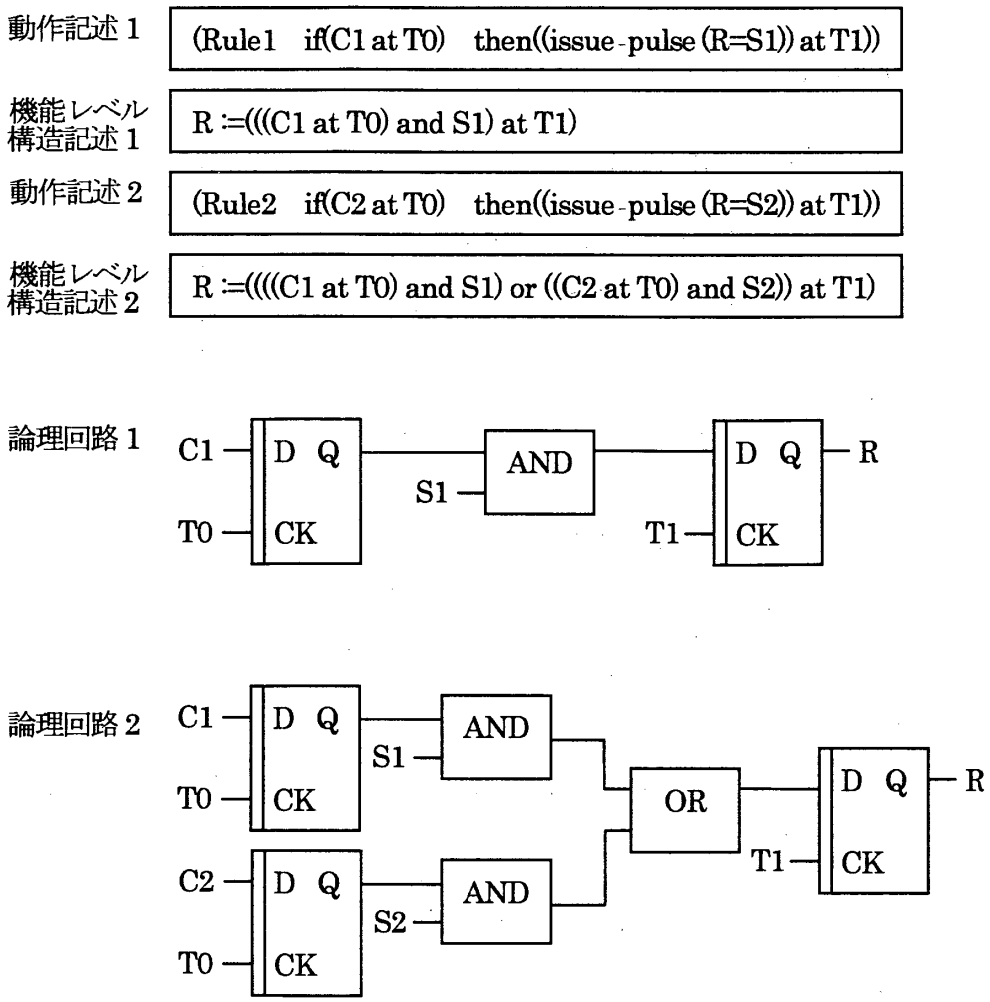

図 5　動作記述の重複

になる。以下にこの手順を述べる。

(1) 機能レベル構造記述の生成

論理動作記述から機能レベル構造記述を生成する。図 4 に論理の記述方法の一例を示す。動作記述は，OR などの組合せ論理に対しては " while( )do( ) " 形式で，Latch を含む順序論理に対しては " if( )then( ) " 形式で記述される。図 4 を見てわかるように，OR の論理であれば OR に対応する構造記述，Latch であれば Latch に対応する構造記述をそれぞれ機械的に生成することができる。

しかし，図 5 に示すように，同一信号名 R に対する動作記述が複数あって重複する場合は，構造記述を機械的に生成すると，論理回路 1 に対応する構造記述を別々に 2 個生成してしまう。これは動作記述では詳細な接続関係が記述しきれていないためである。また，構造記述自体が複数行に渡り，直接 1 個の構造記述にまとめて生成するのが困難なためでもある。本論文では，動作記述を補う膨大な生成規則を用意することが困難なため，動作種別と動作記述に重複があるかないかの別に応じて機能レベル構造記述の生成規則を用意しておくことにする。そうしておけば，まず動作記述 1 から動作記述に重複のない場合の生成規則に従って，

用意した生成パターンに信号名を埋め込むことで，機能レベル構造記述1を生成できる。次に、動作記述2から信号名Rが既に機能レベル構造記述1にあることを判断して、動作記述に重複のある場合の生成規則に従い，用意した生成パターンに信号名を埋め込んで，機能レベル構造記述2を生成できる。最後に機能レベル構造記述1を削除して終了する。このようにして，図5の論理回路2に対応する機能レベル構造記述だけが生成できることになる。

(2) 実装レベルの構造記述変換

図4を見てわかるように，機能レベル構造記述は機械的に構造記述に変換生成することができる。

(3) 最適化

最適化は論理レベルと実装レベルの両方でそれぞれ行った。論理レベルの最適化は機能レベル構造記述における論理式の簡単化を行なう。例えば，(A=1) はAで置き換える，((A at TO) and 1) は (A at TO) と簡単化できる。実装レベルでの最適化については，使用する論理回路によっては，例えばECL回路では肯定と否定の両方の出力が可能であることを利用する。この否定側の出力を利用することにより否定回路を削除できる。

4. 評価

デイレー要素生成部分を試作し，論理生成時のゲート数を評価した。プログラミング言語には，再帰的なデータ構造が扱い易いLISP言語を用いた。計算機論理の中の制御系論理の一部を取り出し，デイレー要素生成の各段階で生成されるゲート数を人手による最適設計結果と比較した。その結果，人手による最適設計では36ゲートになったのに対して，機能レベル構造記述の段階で75ゲート，その論理式の簡単化処理後で49ゲート，論理回路に依存する簡単化処理後で38ゲートであった。したがって，人手による最適設計の場合と比較して，ゲート数の増加は最終的に10%程度に抑えることができ，ある程度実現の見通しが得られた。

5. むすび

計算機を設計するうえで，計算機実装系の緒元を基にした論理設計段階での性能を予測するシステムの構想を試みた。計算機の性能を把握する指標としてマシンサイクルタイムをとりあげた。計算機の論理方式を論理記述ファイルに組み込み，計算機の実装方式を実装系仕様ファイルと評価基準ファイルに組み込む。これらのファイルから論理系と実装系を融合したディレー要素ファイルを生成していく。ディレー要素ファイルの生成には，ディレー要素生成，実装系写像，ディレー計算，パスディレー解析の段階を経るものとし，最終的にマシンサイクルタイムを算出できるように構想した。本論文では，ディレー要素ファイルのフォーマットを階層的な記述で示し，ディレー要素生成の手順を示した。ディレー要素生成段階での論理生成を試みた結果，人手による最適設計の場合に比較して，生成されるゲート数は10%程度の増加に抑えることができる見通しが得られた。

今後は，実装系写像，ディレー計算，パスディレー解析の各段階の処理プログラムを追加してシステムを一通り完成させ，情報教育に応用していきたい。

参考文献

1) J.A.Darringer et al. , "LSS:A System for Production Logic Synthesis," IBM J. Res. Develop. , Vol.28, No.5, pp.537–545, September, 1984.